Learning Deep Nearest Neighbor Representations Using Differentiable Boundary Trees

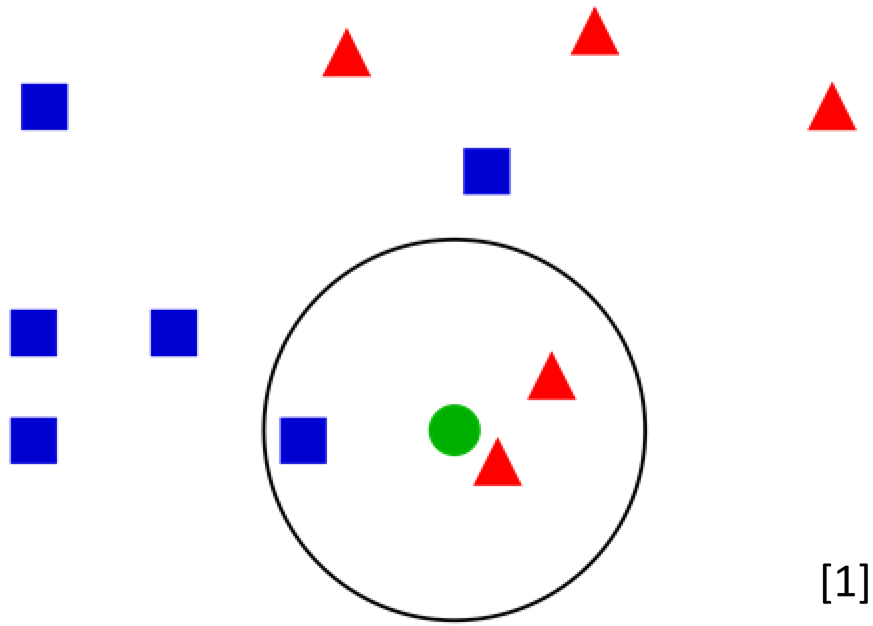
Benedikt Kersjes

Explainable Machine Learning

24.05.2018

Motivation

For classification with k-nearest-neighbour methods we need to find a representation and distance metric.

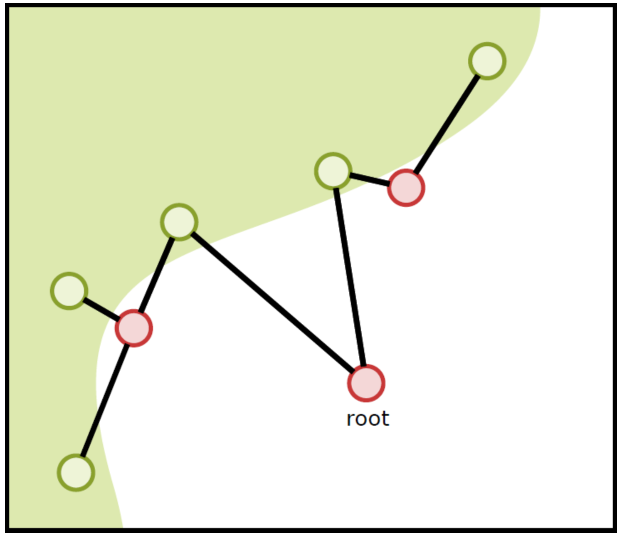


[1]

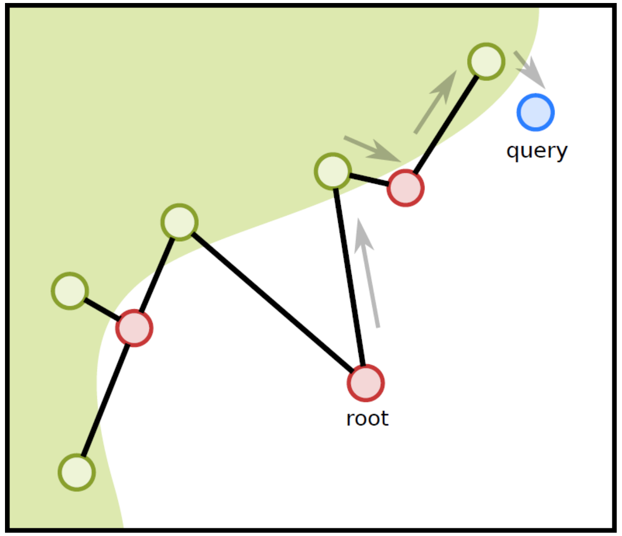
Boundary Trees

Paper: The boundary forest algorithm for online supervised and unsupervised learning. [Mathy, Derbinsky, Bento, Rosenthal 2015]

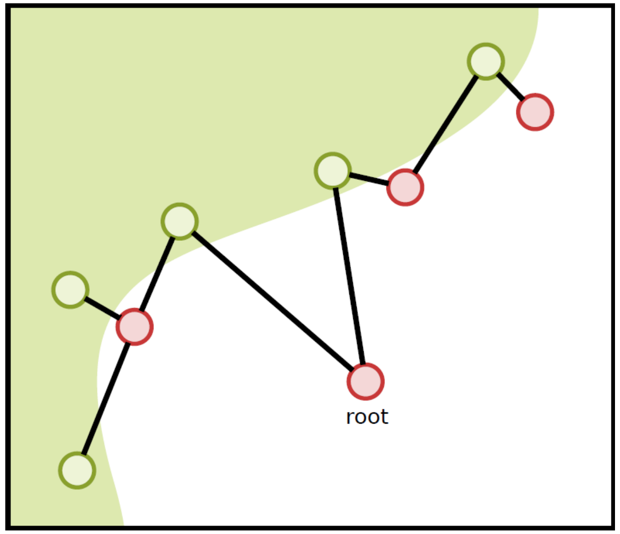
Boundary Trees



Starting tree



Traverse tree until we reach locally closest node. Use this nodes label as prediction.



[2]

If the prediction is correct, discard the query node. Otherwise add it as child to the locally closest node.

Problem?

Algorithm uses raw input representation

Differentiable Boundary Trees

Paper: Learning Deep Nearest Neighbor Representations Using Differentiable Boundary Trees. [Zoran, Lakshminarayanan, Blundell 2017]

Representation for Boundary Trees

Idea: Learn representation for Boundary Trees
→ Simple boundaries in transformed space

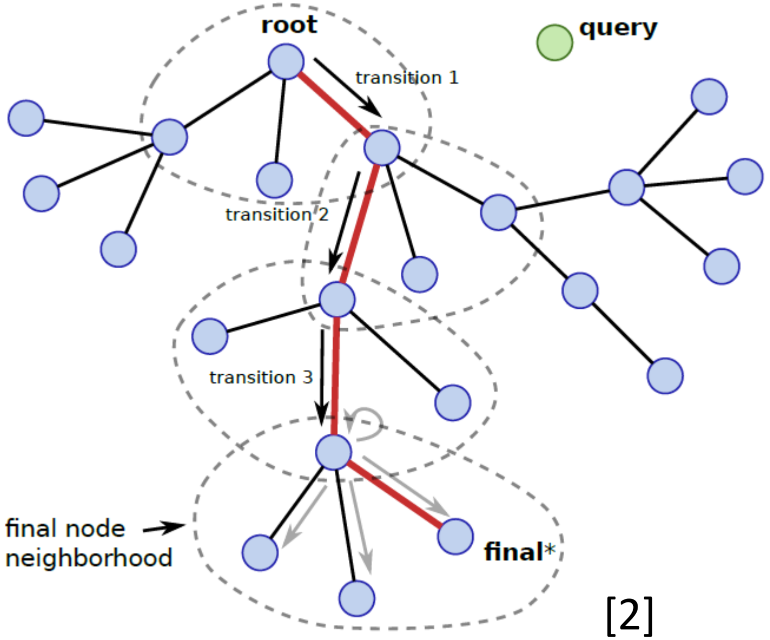
Differentiable Boundary Trees

- Model transitions as stochastic events

$$p(x_i \rightarrow x_j | y) = \underset{i,j \in child(i)}{SoftMax}(-d(x_j, y))$$

- Probability for path from root to final node

$$p(path|y) = \prod_{i \rightarrow j \in path} p(x_i \rightarrow x_j | y)$$



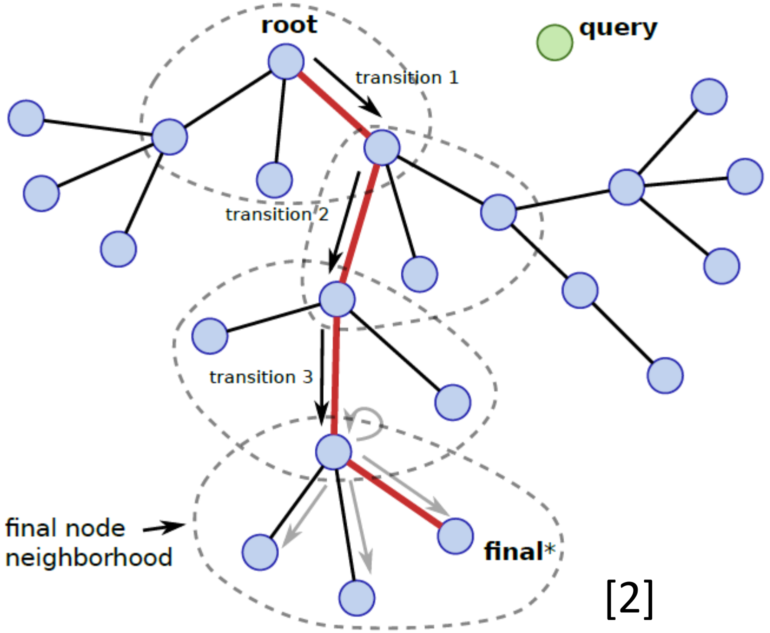
[2]

Differentiable Boundary Trees

- Simplify by taking greedy path

$$p(c|y) = \mathbb{E}_{path|y}(p(c|path, y)) \approx p(c|path^*, y)$$

- Take siblings of final note into account



[2]

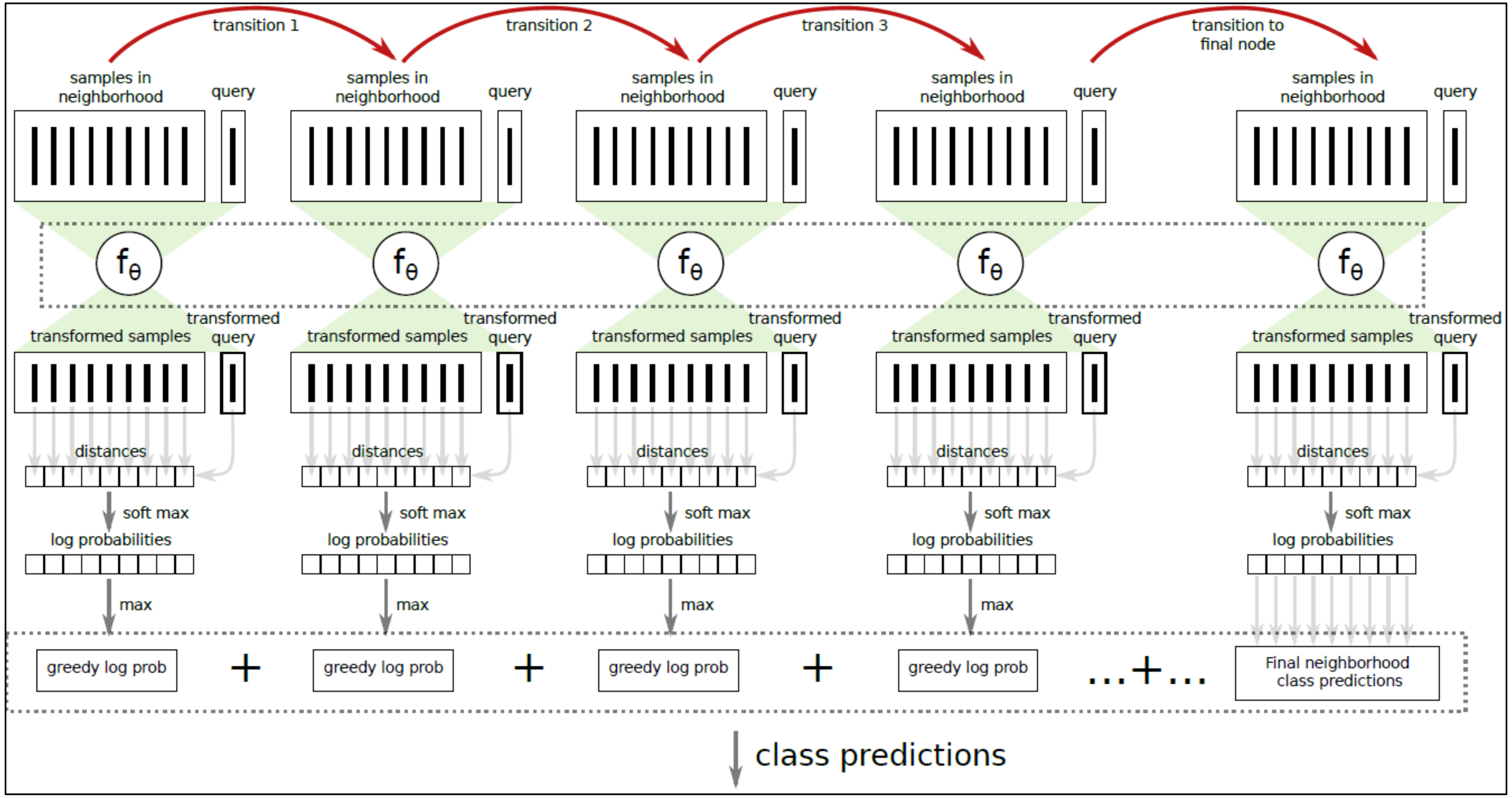
$$log\ p(c|y) = \sum_{x_i \to x_j \in path^+|y} log\ p(x_i \to x_j|y)\ +\ log \sum_{x_k \in sibling(x_{final^*})} p(parent(x_k) \to x_k|y)c(x_k)$$

Differentiable Boundary Trees

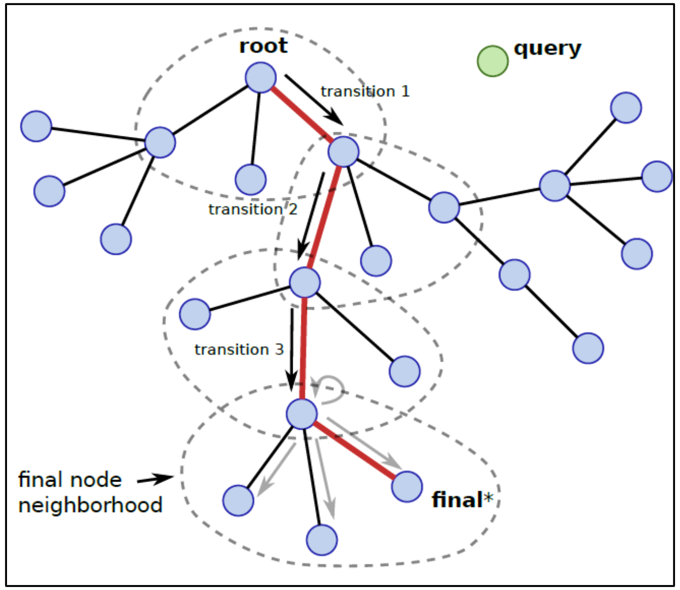
- Apply transformation

$$log\ p(c|f_\theta(y)) = \sum_{x_i \rightarrow x_j \in path^+|f_\theta(y)} log\ p(f_\theta(x_i) \rightarrow f_\theta(x_j)|f_\theta(y))$$
$$+\ log \sum_{x_k \in sibling(x_{final^*})} p(parent(f_\theta(x_k)) \rightarrow f_\theta(x_k)|f_\theta(y))c(c_k)$$

Architecture



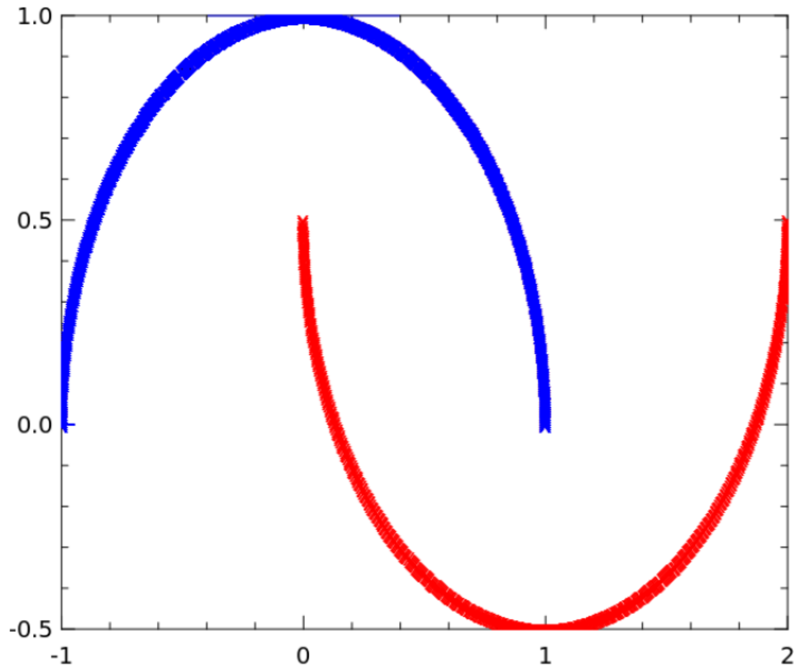
[2]



[2]

Evaluation

Experiment 1: Half-moon dataset

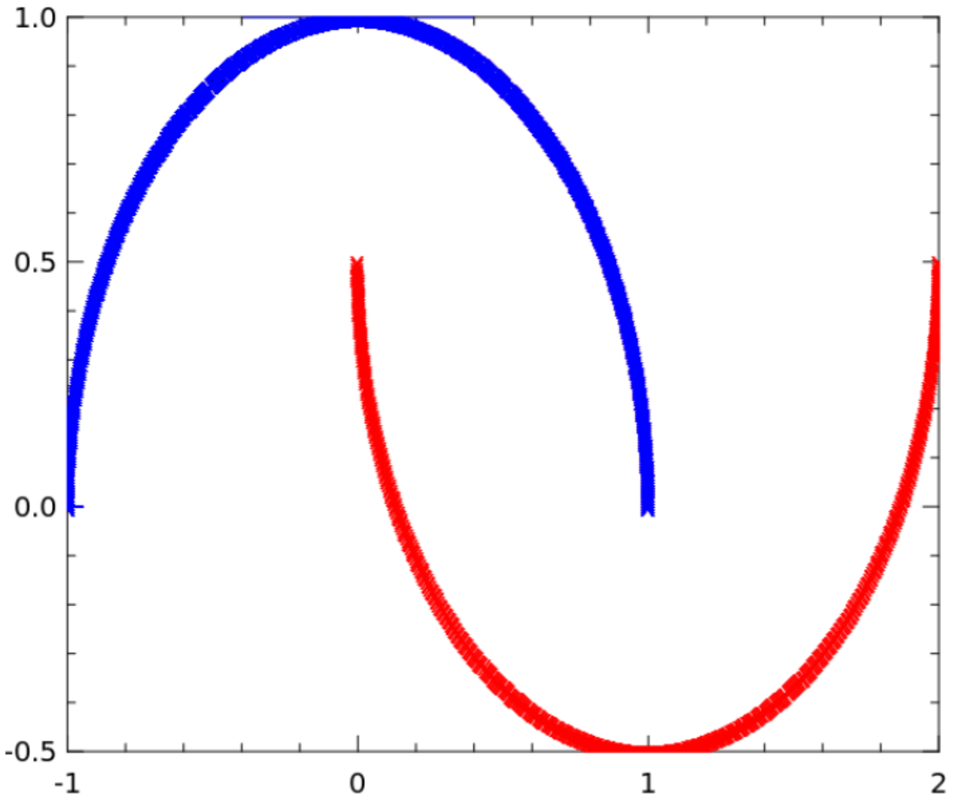


[2]

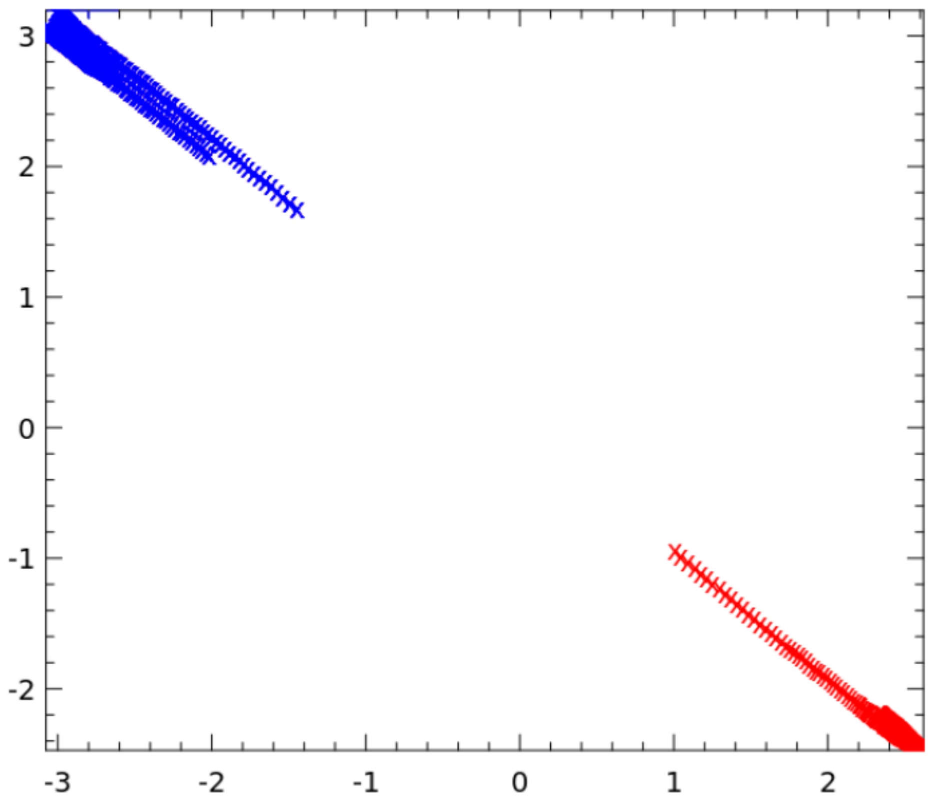
Experiment 2: MNIST classification

[3]

Results – Half moon dataset



Half moon dataset

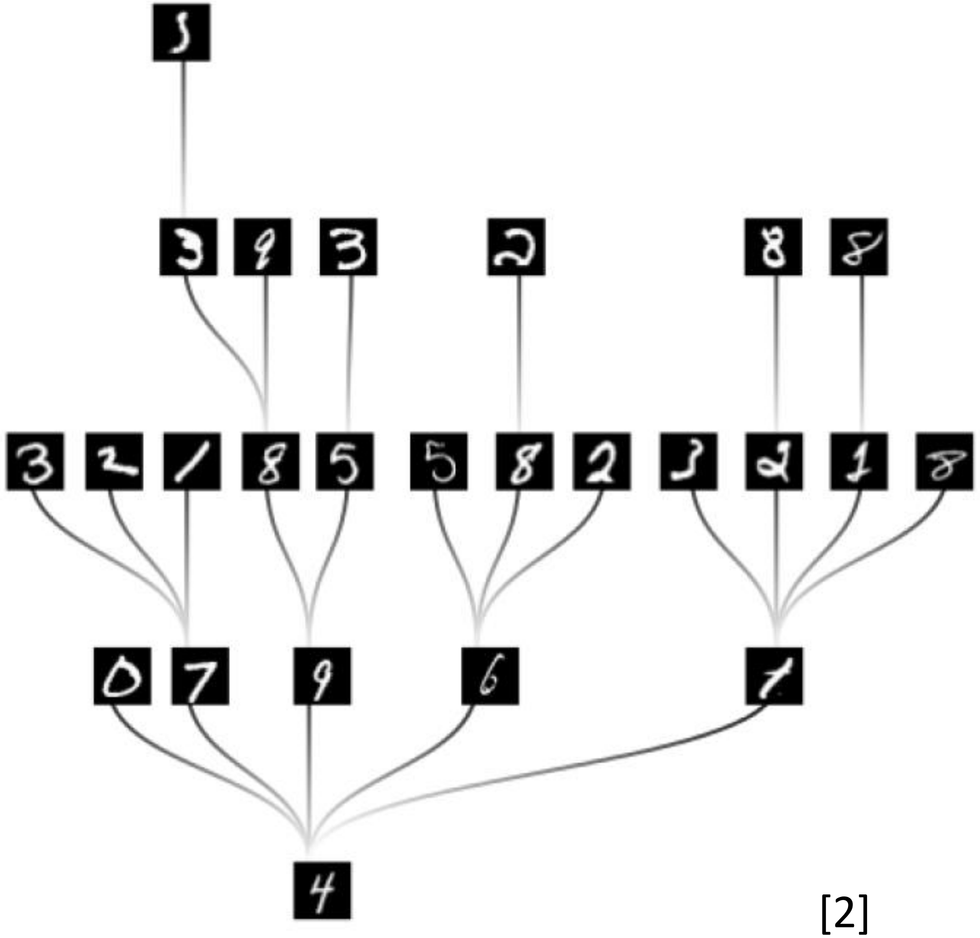


[2]

Learned representation

Results - MNIST

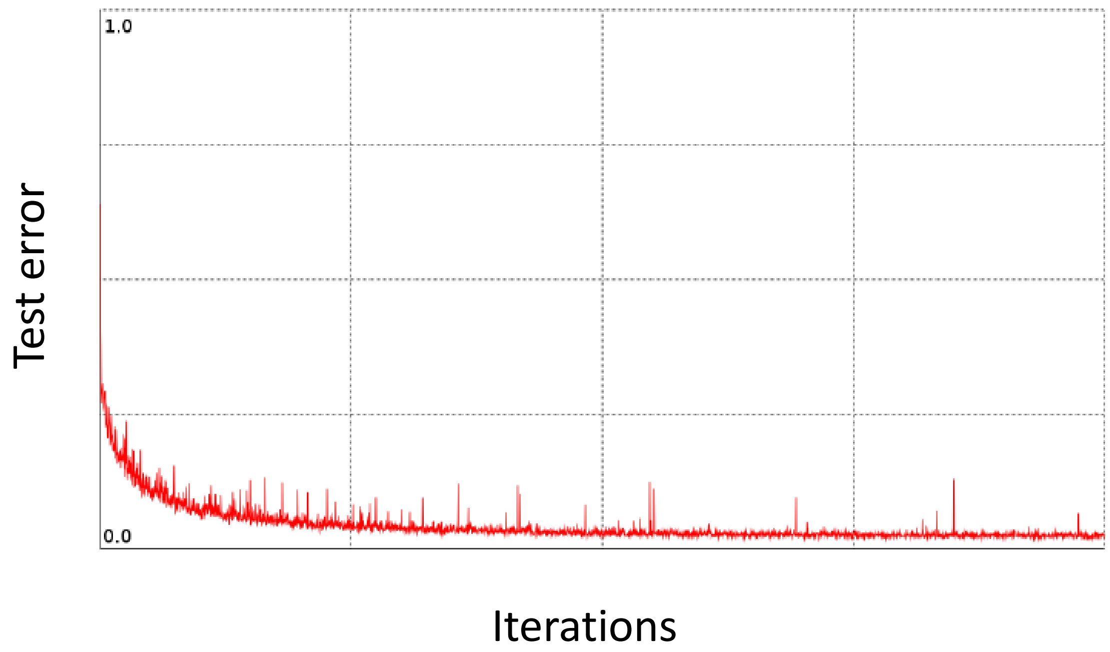
Method	Test Error Rate	Number of nodes
Boundary tree (raw pixels)	11.01%	8536
Boundary tree (pre-trained net)	5.5%	2906
1-NN (Raw pixels)	5.0%	60,000
Neural net (directly as classfier)	2.4%	-
Boundary tree (our learned representation)	**1.85%**	**202**

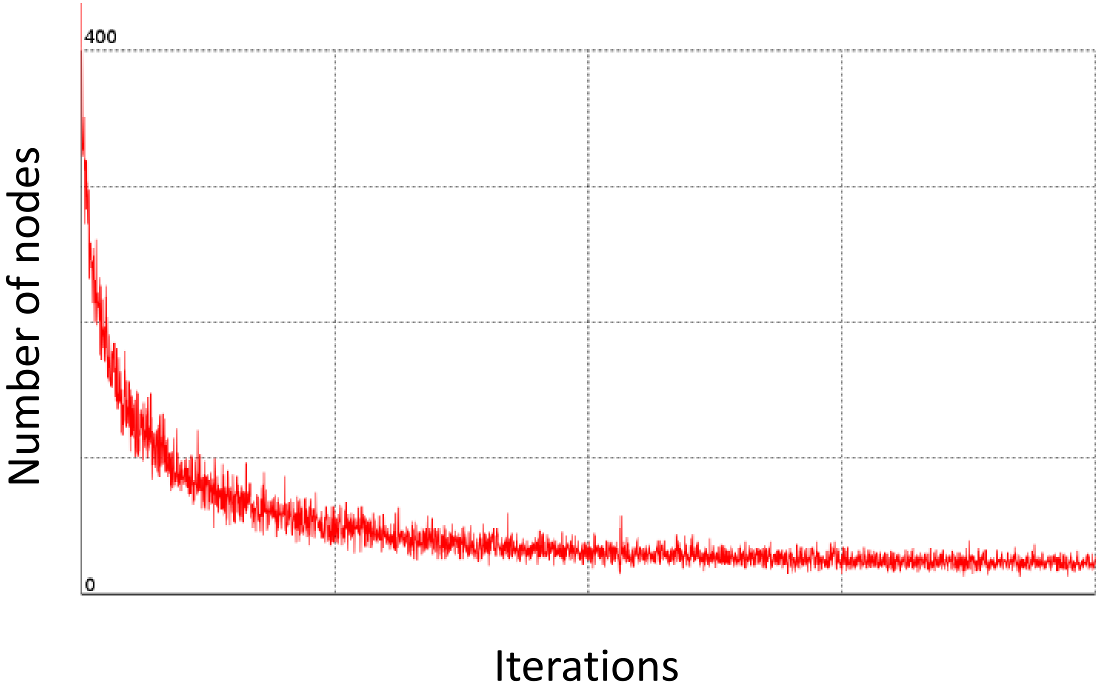


[2]

Resulting Boundary Tree

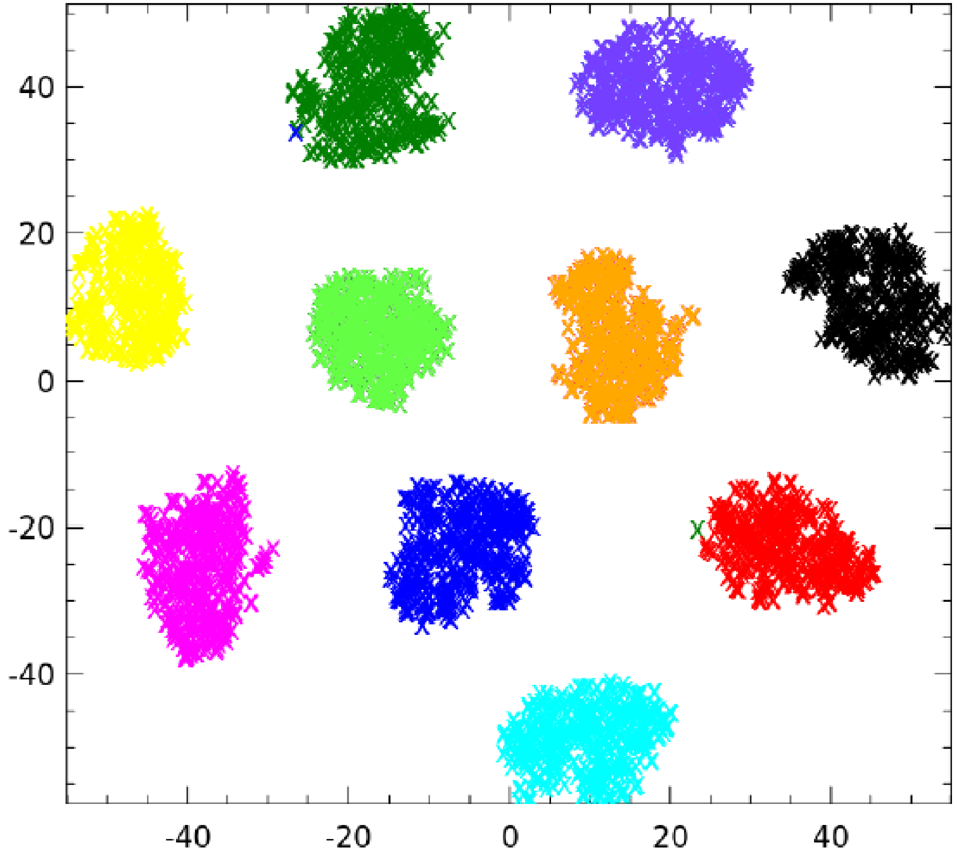
Results - MNIST



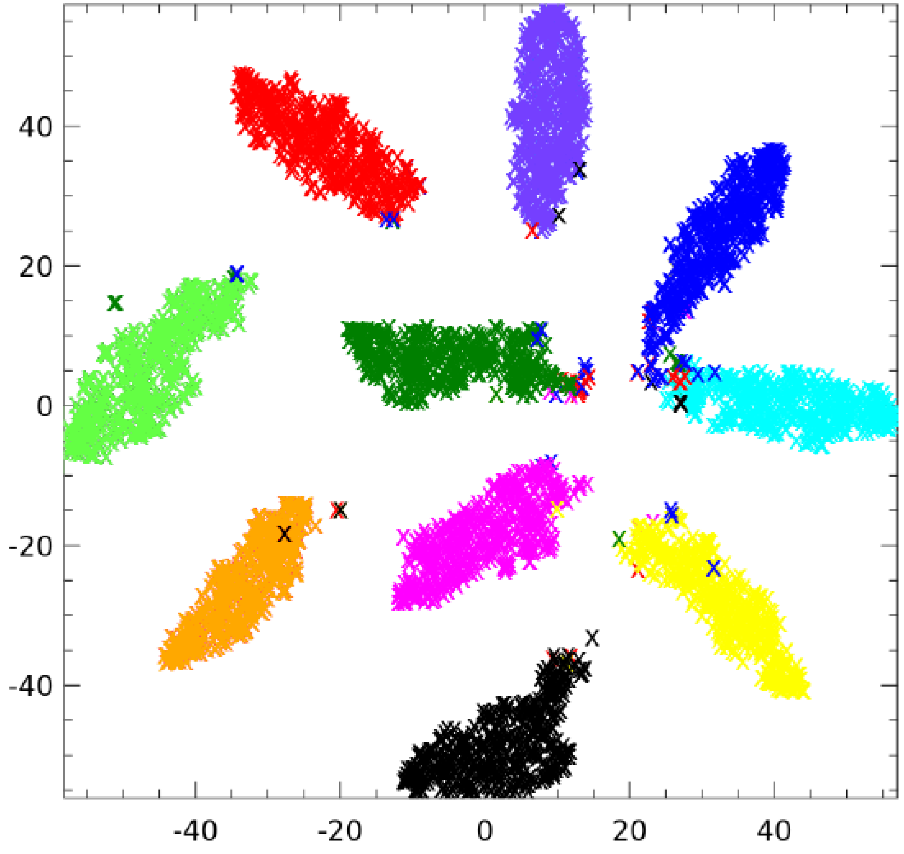


[2]

t-SNE Visualisation



Boundary Tree

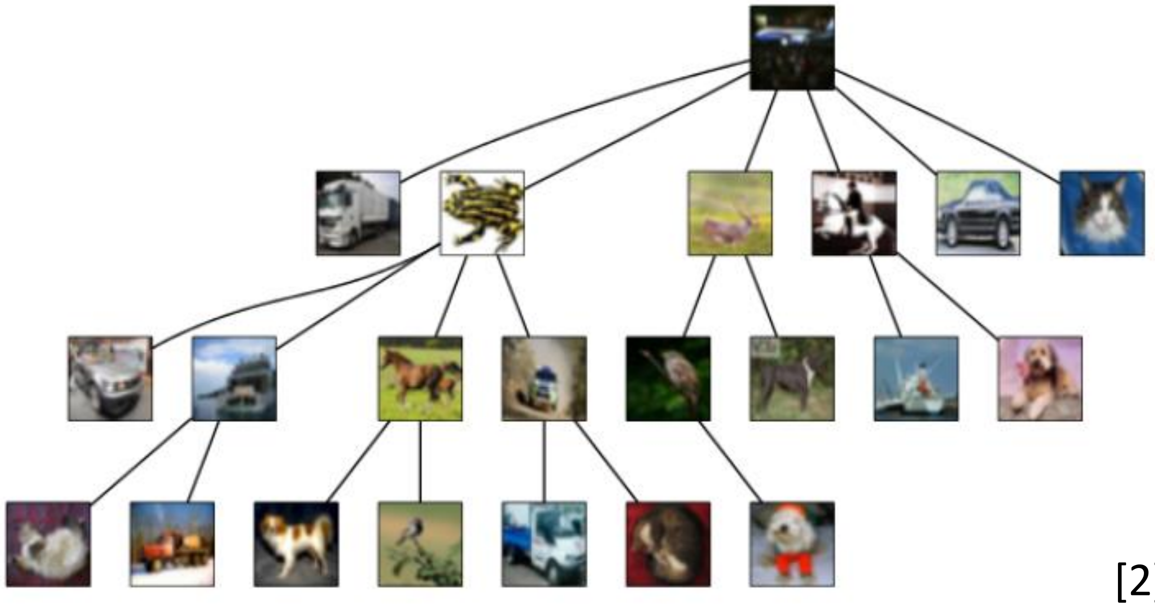


Neural network

[2]

Limitations

- No batching possible
- Large trees in the beginning



[2]

Conclusion

- Simple structure
- Fast queries
- High accuracy

- Does not scale yet

Thank you!

Sources

- [1] https://commons.wikimedia.org/wiki/File:KnnClassification.svg
- [2] Learning Deep Nearest Neighbor Representations Using Differentiable Boundary Trees. [Zoran, Lakshminarayanan, Blundell 2017]
- [3] https://devmesh.intel.com/projects/digit-classifier-using-mnist-dataset-16219